04. Estructuras de control

Programación - 1º DAM

Luis del Moral Martínez

versión 20.10

Contenidos del tema

1. Estructuras de control
2. Estructuras de selección
3. Estructuras de repetición

1. Estructuras de control

¿Qué es una estructura de control?

- Controlan el flujo de ejecución de un programa o una función
- Permiten combinar instrucciones o sentencias individuales
- Existen tres tipos de estructuras de control
  - **Secuencial**: todo el código visto hasta ahora es de tipo secuencial
  - **Selección (decisión)**: ejecución de código en función de una determinada condición
  - **Repetición (iteración)**: repetición de cierto código en función de una determinada condición
- Las estructuras de control utilizan sentencias compuestas (agrupación de sentencias entre { })

2. Estructuras de selección

¿Qué es una estructura de selección?

- Las estructuras de selección permiten ejecutar un código u otro en función de una condición
- Existen dos tipos: **sentencia IF** y **sentencia SWITCH**

2. Estructuras de selección

Sentencia IF

- Supone la estructura de selección principal
- Tiene dos alternativas posibles:
  - **Condición simple**: IF
  - **Condición doble**: IF-ELSE

2. Estructuras de selección

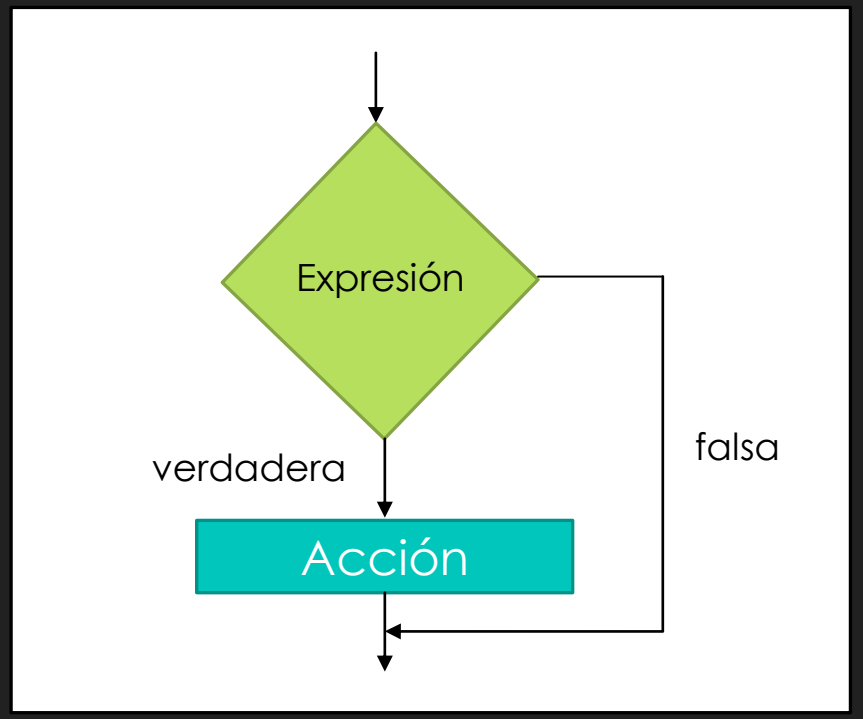
Sentencia IF con condición simple

if (expresión) accion

- Acción que será ejecutada
- Expresión lógica que será evaluada
- Palabra reservada **if**

```
if (calificacion >= 5)
    cout << "El alumno está aprobado" << endl;
```

- Abre el fichero **04_01_sentencia_if_simple.cpp**



2. Estructuras de selección

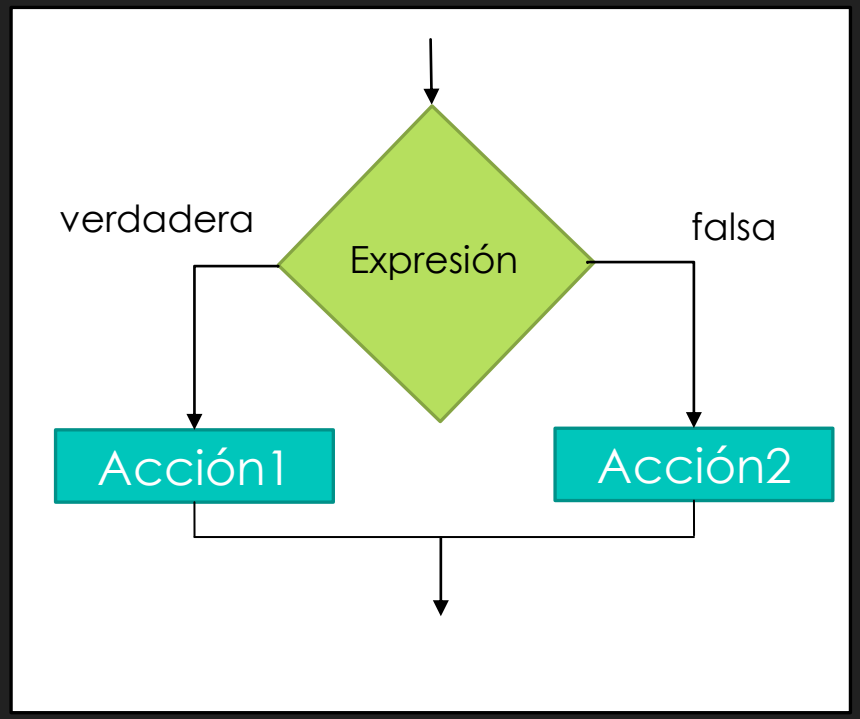
Sentencia IF con condición doble (IF-ELSE)

if (expresión) accion1 else accion2

Acción si la condición es negativa

Acción si la condición es positiva

Expresión lógica que será evaluada




```
if (calificacion >= 5)
    cout << "El alumno está aprobado" << endl;
else
    cout << "El alumno está suspenso" << endl;
```

- Abre los ficheros **04_02_sentencia_if_doble.cpp** y **04_03_sentencia_if_doble_2.cpp**

2. Estructuras de selección

Sentencias IF-ELSE anidadas

- Es posible anidar varias condiciones repitiendo la cláusula else if

if (expresión1) accion1 else if (expresión2) accion2 else if (expresión3) acción3 else acción4

- Al final se puede añadir una cláusula **else** si no se cumple ninguna condición

```
if (x >= 0)
    cout << "El número es mayor o igual a 0" << endl;
else if (x < 0)
    cout << "El número es menor que 0" << endl;
else
    cout << "El número es igual a 0" << endl;
```

- Abre el fichero **04_04_sentencia_if_anidada.cpp**

2. Estructuras de selección

Sentencias IF-ELSE con sentencias complejas

- Es posible incluir sentencias complejas (bloque de código entre { y }
- Esta operativa es válida tanto para el **if simple** como para el doble y el **anidado**

if (expresión1) { acciones } else if (expresión2) { acciones } … else { acciones }

```
if (x <= y)
{
    cout << "Valor de x: " << x << endl;
    cout << "Valor de y: " << y << endl;
    cout << "x es menor o igual que y" << endl;
}
```

- Las expresiones, a su vez, pueden ser compuestas y usar los operadores **relacionales** o **lógicos**
- Abre el fichero **04_05_sentencia_if_sentencias_complejas.cpp**

2. Estructuras de selección

Sentencia SWITCH: condiciones múltiples (1)

- Permite seleccionar una alternativa
- Es útil cuando queremos usar el valor de una variable
- La expresión de selector puede ser **int** o **char**
- Se suele utilizar para hacer **un menú** en el programa

```
switch (selector)
{
   case etiqueta1 : sentencias; break;
   case etiqueta2 : sentencias; break;
...
   default: sentencias; // opcional
}
```

```
switch (x)
{
     case 1:
          cout << "Uno" << endl;
          break;

     case 2:
          cout << "Dos" << endl;
          break;

     case 3:
          cout << "Tres" << endl;
          break

     default:
          cout << "Fuera de rango" << endl;
          break;
}
```

2. Estructuras de selección

Sentencia SWITCH: condiciones múltiples (1)

- Si el valor del selector está en un case:
  - Se ejecutan las sentencias
- Si el valor del selector no está en ningún case:
  - No se ejecuta ninguna sentencia o
  - Se ejecuta la cláusula **default** (opcional)
- La sentencia **break** es importante
  - Sirve para terminar el switch (¡No omitirla!)

```
switch (x)
{
    case 1:
        cout << "Uno" << endl;
        break;

    case 2:
        cout << "Dos" << endl;
        break;

    case 3:
        cout << "Tres" << endl;
        break

    default:
        cout << "Fuera de rango" << endl;
        break;
}
```

- Abre los ficheros **04_06_sentencia_switch.cpp** y **04_07_sentencia_switch_menu.cpp**

3. Estructuras de repetición

¿Qué es una estructura de repetición?

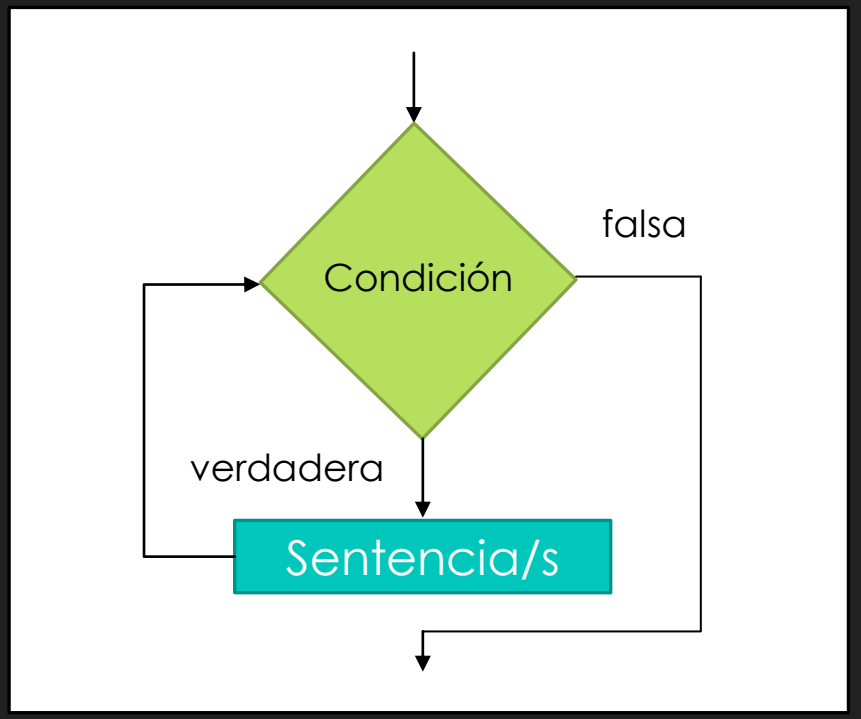
- Las estructuras de repetición repite una secuencia de sentencias un número de veces
- La repetición se lleva a cabo si se cumple una determinada condición
- Existen tres tipos: **sentencia WHILE**, **sentencia DO-WHILE** y **sentencia FOR**

3. Estructuras de repetición

Sentencia WHILE (1)

- El bucle While tiene una condición de parada
- La condición controla la repetición de sentencias
- La condición se evalúa antes de entrar en el bucle

```
while (condición)
{
   sentencias;
}
```



- Abre el fichero **04_08_bucle_while.cpp**

3. Estructuras de repetición

Sentencia WHILE (2)

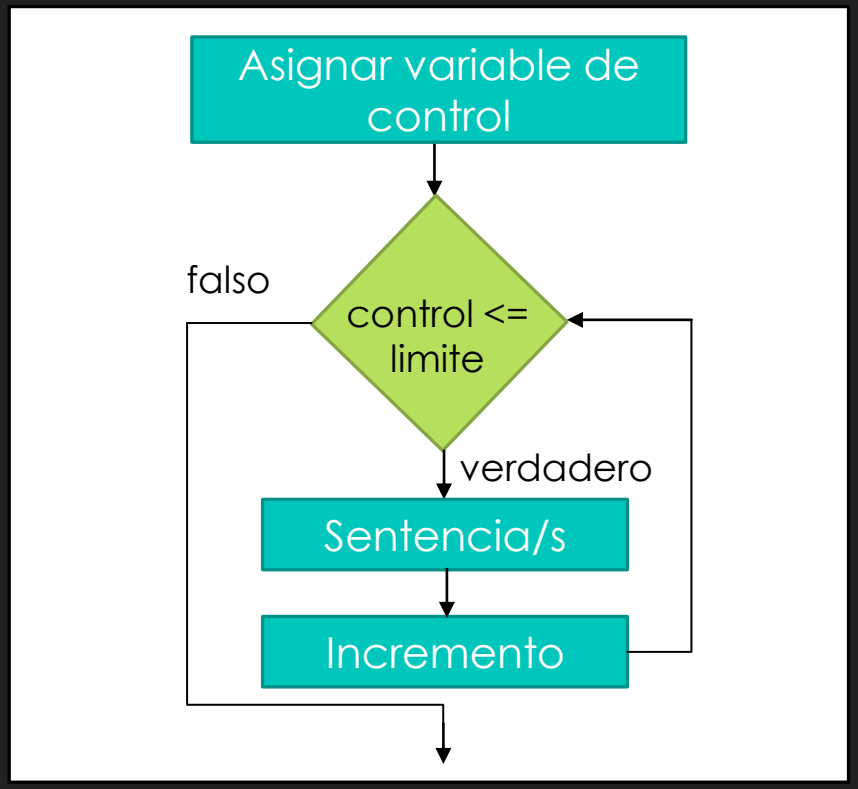
- Consideraciones:
  - El cuerpo del bucle se repite mientras que la **condición sea verdadera**
  - La condición se puede controlar con operadores de **incremento** (++) y **decremento** (--)
  - Si se ejecutarán sentencias complejas, es vital ubicar el código entre llaves **{** y **}**
  - Si no se saben a priori el número de iteraciones, usar un **centinela** en la condición (!= -1 por ejemplo)
  - Se puede controlar el bucle mediante un booleano (un valor de **bandera true** o **false**)
  - Diseñar bien las condiciones, puede provocar un bucle infinito

3. Estructuras de repetición

Sentencia FOR (1)

- Ejecuta un bloque de código un número fijo de veces
- La sentencia for tiene cuatro partes
- Hay que usar ; para separar cada parte de la cabecera

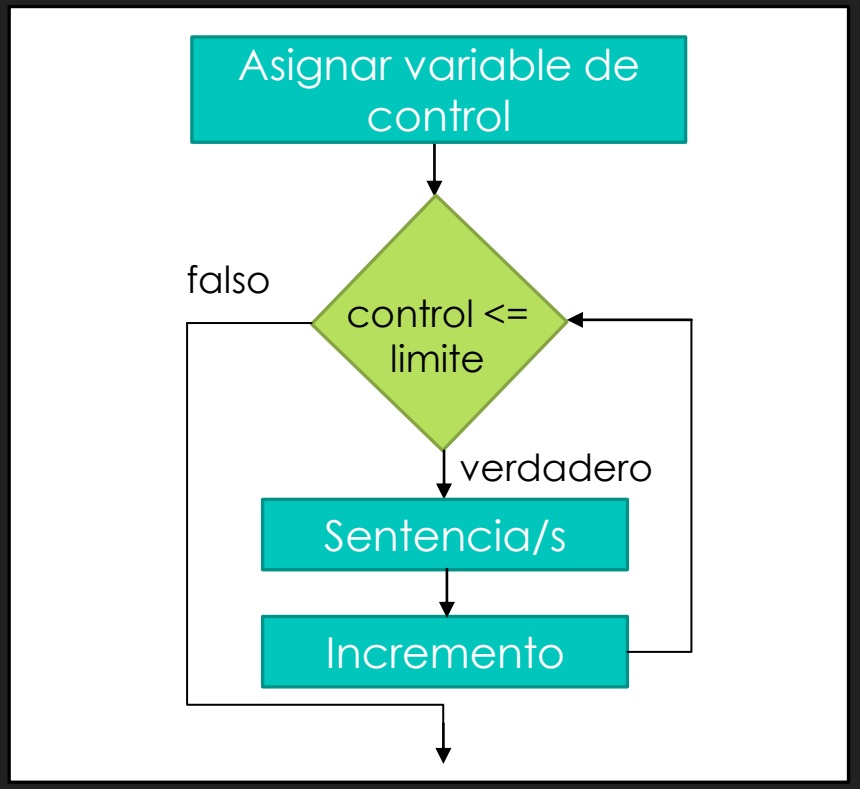
```
for (inicialización; condición; incremento)
{
    sentencias;
}
```



3. Estructuras de repetición

Sentencia FOR (2)

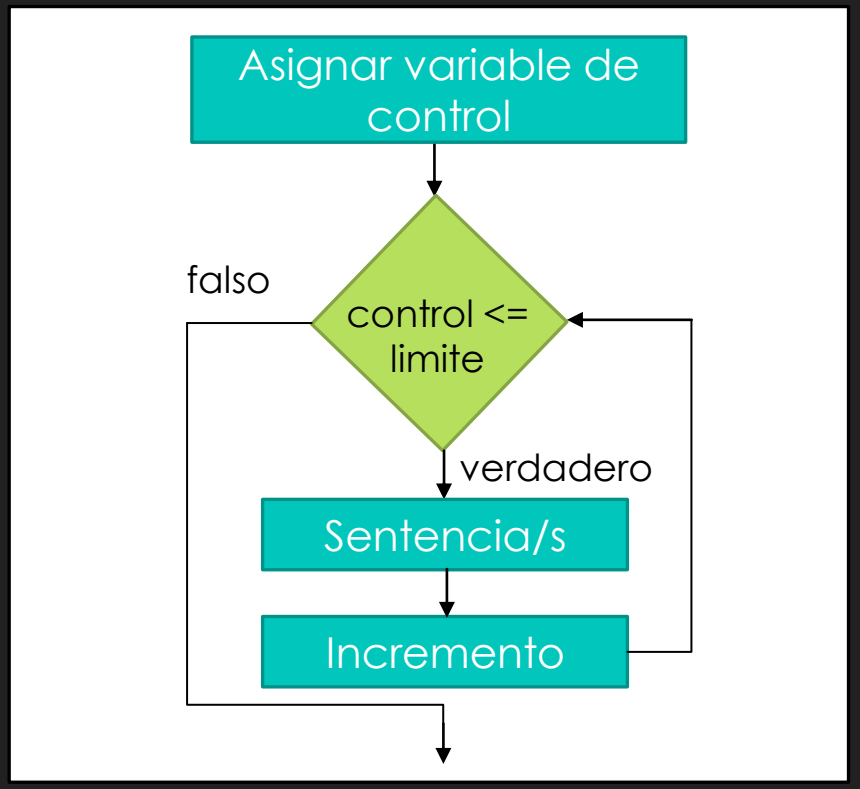
- **Parte de inicialización**:
  - Inicializa las variables de control del bucle
  - Podemos usar una variable declarada previamente
  - Podemos declarar la variable en el bucle (int i = 0;)
- **Parte de condición**:
  - Contiene una expresión lógica
  - Si es verdadera, se realizan las iteraciones de las sentencias
- **Parte de incremento**: incrementa la variable de control



3. Estructuras de repetición

Sentencia FOR (3)

- **Precauciones en el uso del FOR**:
  - Cuidado si modificamos las variables de control del bucle
  - Podemos incrementar las iteraciones
  - El bucle puede derivar en un bucle infinito



- Abre los ficheros **04_09_bucle_for.cpp** y **04_10_bucle_for_2.cpp**

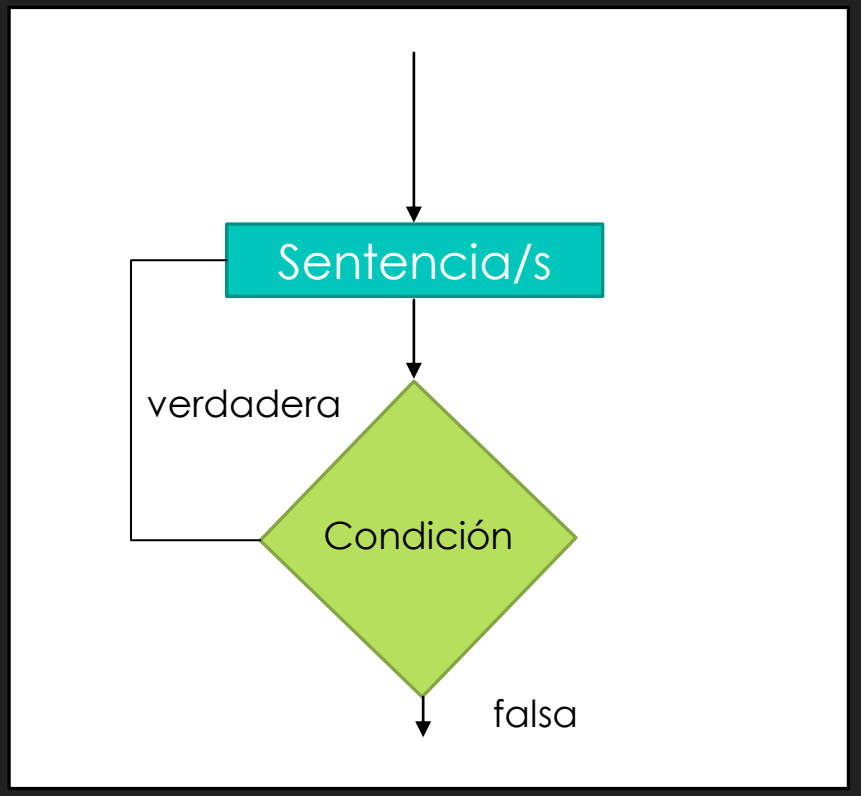
3. Estructuras de repetición

Sentencia DO-WHILE

- El bucle Do-While tiene una condición de parada
- El bucle se ejecuta al menos una vez
- La condición se evalúa después de la primera iteración

```
do
{
   sentencias;
} while (condición)
```

- Abre el fichero **04_11_bucle_do_while.cpp**



3. Estructuras de repetición

¿Cuándo conviene utilizar cada bucle?

Bucle	Uso
while	Cuando la repetición no está controlada por un contador, sino por una condición. El cuerpo del bucle puede no ser ejecutado, en función de esta condición
for	Cuando se conoce el número de repeticiones de manera anticipada, y puede ser controlado por un contador
do-while	Cuando queremos que el bucle se ejecute, al menos, una vez

3. Estructuras de repetición

Ruptura de control en los bucles

- Podemos abandonar o salir de una iteración para **saltar** partes del bucle
- Existen tres sentencias para hacer esto: **break**, **continue** y **goto** (no usar JAMÁS)
  - **break**: transfiere el control del programa final del bucle (en un switch termina la selección)
  - **continue** (su uso es menos frecuente que break)
    - **En bucle while**: retorna a la expresión de control del bucle
    - **En bucle for**: vuelve a la tercera expresión del bucle for, saltándose el resto del bucle
- En cualquier caso, no resulta recomendable usar estas instrucciones
- Complican la comprensión de los bucles (y pueden provocar errores lógicos)

3. Estructuras de repetición

Bucles anidados

- Existe la posibilidad de **anidar** un bucle dentro de otro
- Con cada iteración de un bucle externo, se repite el bucle interno por completo
- Hay que tener precaución a la hora de definir los nombres de las variables de control
- En el siguiente código, el **cout** se ejecutará **5x5 = 25 veces**

```
for (int i = 0; i < 5; i++)
    for (int j = 0; j < 5; j++)
        cout << i*j << endl;
```

Créditos de las imágenes y figuras

Cliparts e iconos

- **Obtenidos mediante la herramienta web IconFinder** (según sus disposiciones):
  - Diapositiva 1
  - Según la plataforma IconFinder, dicho material puede usarse libremente (free comercial use)
  - A fecha de edición de este material, todos los cliparts son free for comercial use (sin restricciones)

Resto de diagramas y gráficas

- Se han desarrollado en PowerPoint y se han incrustado en esta presentación
- Todos estos materiales se han desarrollado por el autor
  - Si se ha empleado algún icono externo, este se rige según lo expresado anteriormente